The Visual Computer manuscript No.
(will be inserted by the editor)

Minh X. Nguyen · Xiaoru Yuan · Baoquan Chen

Geometry Completion and Detail Generation by Texture Synthesis

Abstract We present a novel method for patching holes in polygonal meshes and synthesizing surface with details based on existing geometry. The most novel feature of our proposed method is to transform the 3D geometry synthesis problem into a 2D domain by parameterizing surfaces and solve this problem in that domain. We then derive local geometry gradient images that encode intrinsic local geometry properties, which are invariant to object translation and rotation. The 3D geometry of holes is then reconstructed from synthesized local gradient images. This method can be extended to execute other mesh editing operations such as geometry detail transfer or synthesis. The resulting major benefits of performing geometry synthesis in 2D are more flexible and robust control, better leverage of the wealth of current 2D image completion methods and greater efficiency.

Keywords Detail preservation · Geometry image and Mesh editing · Partial differential equation · Surface completion

1 Introduction

Holes are considered to be major deficiencies in polygonal meshes themselves. Holes can be introduced by some mesh editing operations. A majority of holes come from polygonal meshes that are obtained from 3D scan data. Polygonal meshes obtained this way are prone to holes due to inter-object and self-occlusions during the scanning process. Moreover, certain physical properties of surface materials (e.g. high reflectivity) may also prevent scanning devices from capturing geometry except from an optimal position. Thus it is desirable to develop a user-friendly mesh editing tool that is capable of automatically filling holes that satisfies the following two requirements:

(1) *Boundary condition*: The patched geometry should match with the existing geometry seamlessly, especially at the boundaries.

(2) *Context condition*: The patched geometry should contain geometry details similar to those on the existing surfaces. For example, a missing region on a brick building wall should be patched with geometry that has a similar relief pattern as the rest of the wall.

Most existing methods satisfy only the boundary condition. The patched geometry is normally smooth and does not convey details i.e., the context condition is not fulfilled. These methods are also constrained in the size of holes they can fix. Recently, Sharf et al. [25] developed a context sensitive surface completion method that synthesizes geometry details by matching and copying 3D structures in the existing geometry.

In this paper, we present a novel approach for patching geometry while synthesizing geometry details so as to fulfill both the boundary and context conditions. Unlike Sharf et al.'s method [25] which represents intrinsic geometry properties and performs geometry completion directly in the 3D domain, we perform these operations in a 2D domain. We unfold the surface mesh into this 2D domain by using a conformal parameterization algorithm. We then compute intrinsic geometry properties representing local geometry variance. After that, we synthesize new geometry variances in the target regions from which the new geometry is constructed. This hole patching method can be extended to other mesh editing operations such as geometry detail transfer or synthesis. By converting the 3D geometry completion problem into a 2D one, we are able to directly utilize a wealth of currently available image completion techniques, achieve more robust user control over the geometry synthesis process, and achieve greater efficiency.

The remainder of the paper is organized as follows. In Section 2, we briefly review some related work. We give

Minh X. Nguyen, Xiaoru Yuan, Baoquan Chen
Department of Computer Science and Engineering
University of Minnesota at Twin Cities
200 Union Street SE
Minneapolis, MN 55455
USA
Tel.: +1-612-625-0365
Fax: +1-612-625-2002
E-mail: {mnguyen,xyuan,baoquan}@cs.umn.edu

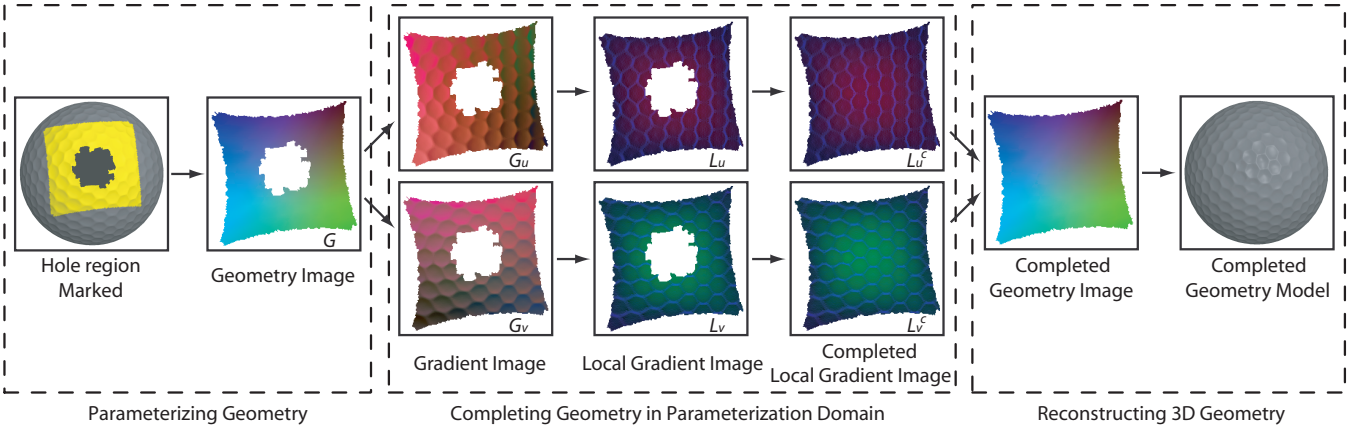


Fig. 1 Work flow of patching a hole with geometry details. A region of geometry surrounding the hole is selected and parameterized into a geometry image (left box). The gradients of the geometry image are computed and transformed into local coordinate systems, forming local gradient images that are completed by 2D texture synthesis methods (middle box). The geometry image of the hole region is finally reconstructed by iteratively solving a PDE followed by a remeshing operation to fit with the surrounding mesh (right box).

an overview of our method in Section 3, and describe each step in detail in Sections 4, 5, and 6. We extend our method to accommodate holes containing islands in Section 7. We discuss results in Section 8, and conclude the paper in Section 9.

2 Related Work

Most previous work on hole filling has focused on producing smooth geometry and satisfying the boundary condition. Pfeifle and Seidel [23] patch holes by using triangular B-splines. Similarly, Liepa [21] patches holes by triangulating hole boundaries in 3D, followed by a fairing process. Levy [19] fills holes by extrapolating geometry boundaries in the parameterization domain by optimizing an energy function. Davis et al. [6] patch holes by locally diffusing a volumetric signed distance field representing the geometry mesh. Ju [17] uses an octree grid in combination with a sign-generation function at each node to repair huge polygonal models. Verdera et al. [28] extend the image inpainting technique introduced by Bertalmio et al. [3] to 3D geometry meshes by extrapolating a PDE from the known regions into the hole regions, and then reconstructing missing holes by solving the established PDE. In all of these methods, whether performed directly on the geometry mesh or in another domain, the resulting geometry in hole regions satisfies only the boundary condition.

In contrast to the above methods, Sharf et al. [25] extend texture synthesis techniques from 2D to 3D for geometry completion to fulfill the context condition as well. Their method first transforms a geometry mesh into an octree. Then recursively from the coarsest level to the finest level of the octree, holes are heuristically determined and their geometry is synthesized using similar geometry information from other cells on the same tree level, much like a 3D version of texture synthesis. However, this method works only with point-based models.

For the detail generation and transfer, recently Bhat et al. [4] used a variation of 3D texture synthesis to modify geometry meshes. Specifically, 3D geometry details are represented as 3D volumetric data and then replicated over the target geometry based on a predefined 3D vector field. More recently, Lai et al. [18] proposed another approach that works for meshes and uses displacement maps to encode geometry details. However, for surfaces of arbitrary curvatures, the synthesized surfaces may penetrate each other due to synthesized large displacements. Special ad-hoc measures are taken to avoid such self-intersection. Since our method uses local gradient images to encode geometry details and then resorts to an iterative PDE solver to reconstruct the new geometry, a globally optimal geometry can be automatically generated.

A distinct difference between our work and Sharf et al.'s method [25] is that we move the difficult task of determining the geometry similarities from the 3D domain into the 2D domain through geometry parameterization. In this 2D domain, local geometry variance is computed. Now that the original 3D problem becomes a 2D one, the boundary and the context conditions are efficiently achieved by utilizing a wide variety of available image completion techniques [2,29]. With geometry intermittently represented in the 2D domain, our method also provides more flexible and robust control over geometry synthesis. Based on the reconstructed geometry image, we reconstruct the connectivity of the mesh based on an analysis of the structure of the original mesh. Note that even though Levy's method [19] also fills holes in a 2D parameterization domain, it generates a smooth mesh and guarantees only the boundary condition.

3 Overview

Once a hole is identified, the user first selects the surrounding geometry of the hole to guide geometry completion. To better assist the user in selecting the region of interest, we implement the "intelligent scissoring" interface presented by Funkhouser et al. in [10] for its effectiveness and robustness. The region of interest includes sample geometry details that the user wants to transfer to target regions. For the moment, we assume that the selected geometry consists of just one connected component. We relax this condition later in Section 7. As illustrated in Figure 1, our geometry completion method consists of the following three steps:

(1) *Geometry Parameterization* (Section 4): For this task, we employ a global least distortion parameterization method [8,12,13,16,20]. After the parameterization, the input mesh is then re-sampled into a geometry image [11]. The pixel colors of the geometry image encode a 3D coordinate mesh.

(2) *Synthesis of Local Geometry Variance* (Section 5): We compute the geometry gradients of the geometry image and transform them to their local coordinates to obtain local gradient images. The local gradient images of hole regions are synthesized using texture synthesis techniques based on local gradient images surrounding the hole.

(3) *Reconstruction of 3D Geometry* (Section 6): The geometry image of the hole regions is reconstructed from the completed local gradient images. After that, meshes for the hole regions are generated based on the constructed geometry image. To ensure that connectivity of the reconstructed mesh conforms with the existing mesh, the geometry mesh is constructed using a constrained conformal Delaunay triangulation [26] in the geometry image domain.

Note that our hole patching method can be extended to performing other mesh editing operations such as geometry detail transfer and synthesis. The geometry details of a source mesh, represented in local gradient images, can be transferred to a destination mesh using the process similar to above described. Here, the local gradient images extracted from the source mesh are used for texture synthesis instead of the local gradient images surrounding the region of the destination mesh to be edited. We provide further discussion about this operation in Section 6.1.

4 Geometry Parameterization

The parameterization step is vital for our method, as it allows us to work with more complex geometry or geometry that is not equivalent to a height field. Moreover, this parameterization is necessary to create geometry images, which are another equivalent representation of geometry.

Our method can work with various parameterization algorithms. The main concern is to find a mapping technique that minimizes the distortion that occurs when a 3D mesh is mapped into the parameterization domain. We choose the global conformal parameterization method presented in [13] because it can handle arbitrary geometry topologies with minimum distortion. This method establishes a system of constraints of parameterization values on the gradient fields of the input geometry mesh, which is formulated as a large sparse least squares linear system. This least squares linear system is then solved using an iterative method. To gain better efficiency, we utilize the ILQ preconditioning method [24] to accelerate convergence.

After the mesh parameterization is computed, we generate the corresponding geometry image. The geometry image is created by drawing all triangles of the input mesh into a 2D domain, where the 2D coordinate of each vertex is its parameterization value, and the color is its 3D coordinate (see the left box of Figure 1).

One issue to consider in this process is the resolution of the geometry image. If the geometry image resolution is not high enough, highly curved shapes in the geometry may be missed due to undersampling. For this purpose, with the assumption that the vertices of geometry mesh are distributed uniformly, we usually create an $m \times m$ geometry image where $m = \lceil\, c\sqrt{n}\,\rceil$, n being the number of vertices of the selected region and c being a constant that the user can specify. In practice, we set c to 2.5, which is often sufficient for both quality and speed.

Parameterization algorithms in general require that the input mesh be connected. They may fail to generate a unique solution or find a reliable solution if the geometry mesh has several disconnected components. We address this issue in Section 7.

5 Synthesis of Local Geometry Variance

From the obtained geometry image, we further compute so-called "local geometry variance images" that describe coordinate-independent geometry patterns. These images are then completed by texture synthesis methods.

5.1 Local geometry variance images

Geometry images do not directly exhibit any intrinsic geometry properties upon which the texture synthesis techniques can be executed. We choose geometry gradients to represent intrinsic geometry properties, but instead of directly using gradient vectors, we transform them into local coordinate systems.

We define $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathcal{C}(\overrightarrow{u}, \overrightarrow{v}, \overrightarrow{N})$ to be the orthonormal coordinate system with major axes $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ formed by vectors $\overrightarrow{u}$, $\overrightarrow{v}$ and $\overrightarrow{N}$. Specifically, $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ are computed as following:

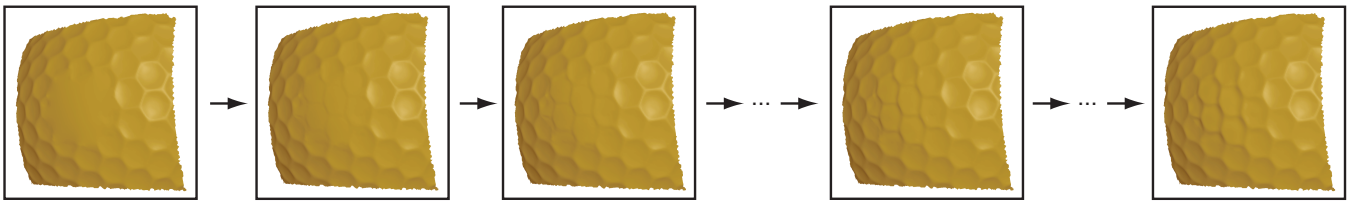


Fig. 2 Geometry construction from local gradient images. The first image is the initialized smooth surface of the hole. The surface is refined iteratively. The last image shows the result after 9 iterations.

$$\mathbf{z} = \frac{\overrightarrow{N}}{|\overrightarrow{N}|}, \mathbf{y} = \frac{\overrightarrow{N} \times \overrightarrow{u}}{|\overrightarrow{N} \times \overrightarrow{u}|} \text{ and } \mathbf{x} = \frac{\mathbf{y} \times \mathbf{z}}{|\mathbf{y} \times \mathbf{z}|} \tag{1}$$

Given a parametric surface F, each point on F has a normal $\overrightarrow{N}$ and gradients

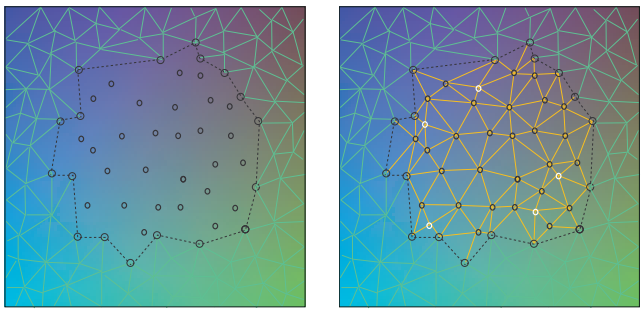
$$\overrightarrow{F_u} = \frac{\partial F}{\partial u} \text{ and } \overrightarrow{F_v} = \frac{\partial F}{\partial v} \tag{2}$$

Note that in the discrete case, $\overrightarrow{N}$ may not be perpendicular to $\overrightarrow{F_u}$ and/or $\overrightarrow{F_v}$. We form a local coordinate system $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathcal{C}(\overrightarrow{F_u}, \overrightarrow{F_v}, \overrightarrow{N})$ based on our prior definition and then transform $\overrightarrow{F_u}$ and $\overrightarrow{F_v}$ into this coordinate system to obtain local gradients. These local gradient vectors are stored into *local gradient images*, which represent local geometry variances. (See Figure 1 - middle box, center column). A local gradient at a given point in the geometry mesh represents its intrinsic geometry property and is invariant to rotations and translations. Therefore, local gradients can be searched and matched between different points on the surface. This local gradient representation and the ability to reconstruct a mesh from it pave a way for achieving a globally optimal solution for many other geometry editing and transferring operations.

Curvature is another geometry property that can be used instead of local gradients. However, curvature is generally complicated to calculate and very sensitive to mesh structure, especially for discrete meshes [27]. On the other hand, normals and gradients can be computed quickly and efficiently in both continuous and discrete domains, and are closely related to curvature [5].

5.2 Completion of local geometry variance images

We rely on texture synthesis methods [2,29] to fill holes in local gradient images. An issue that we address here is the non-uniform distortion (i.e., scaling) of geometry parameterization and hence the local gradient images. The hole pixels have unknown scaling factors. Therefore, we need to generate local gradient images of different scales for the input geometry so that we can find a best match for each pixel to be synthesized. We create multiple image copies of the input local gradient images by up-scaling and down-scaling them s times, each time by $\sqrt{2}$ (s is a user defined constant depending on the geometry characteristics, usually $1 < s < 4$ is sufficient for most cases). Another issue is the orientation of local gradients. This orientation is described by a rotation within the 2D image domain. We further generate r copies of each input image of a specific scale, where r is another user defined constant. Specifically, each image k ($k \in [0, r)$) is rotated by an angle $k * 360/r$. Now, with rs images provided as the input textures, we perform conventional texture synthesis.



(a) (b)

Fig. 3 Remeshing of hole geometry: (a) high feature points (black) are selected according to the importance map, and (b) a constrained Delaunay triangulation is applied (additional points, colored white, are added). The boundary of the hole is marked by circular dots and dashed lines.

6 Reconstruction of the 3D Geometry

Next we reconstruct 3D geometry from the completed local gradient images. The reconstruction process consists of two parts: recovering the geometry image from the completed local gradient images and remeshing geometry for the hole regions.

6.1 Reconstructing the geometry image

Geometry images for the hole regions cannot be directly reconstructed from local gradient images because they

are defined in local coordinate systems. We need to first transform these local gradients into a global coordinate system to obtain global gradient images from which a geometry image can be reconstructed. We employ an iterative approximation approach to compute the complete geometry image. We initialize a geometry image and then refine it based on the synthesized local gradient images in the hole regions.

We denote the incomplete geometry image as G, its global gradient images as G_u and G_v, the completed local gradient images as L_u^c and L_v^c (see Figure 1). Our iterative process is summarized in the following pseudo-code (steps 1 and 2 perform initialization of smooth patch):

1: Patch G_u by linearly interpolating its values along the u direction to get G_u^c.
2: Patch G_v by linearly interpolating its values along the v direction to get G_v^c.
3: Reconstruct geometry image G^0 from global gradient images G_u^c and G_v^c (Appendix A)
4: $i \leftarrow 0$
5: $error \leftarrow \infty$
6: **while** $error > threshold$ **do**
7: Compute global gradient image G_u^i from G
8: Compute global gradient image G_v^i from G
9: Compute normal map G_N^i from G
10: $error \leftarrow 0$
11: **for** hole pixel (s,t) in G **do**
12: $\overrightarrow{N} \leftarrow G_N^i(s,t)$
13: $\overrightarrow{F_u} \leftarrow G_u^i(s,t)$
14: $\overrightarrow{F_v} \leftarrow G_v^i(s,t)$
15: $\overrightarrow{l_u} \leftarrow L_u^c(s,t)$
16: $\overrightarrow{l_v} \leftarrow L_v^c(s,t)$
17: $(\mathbf{x},\mathbf{y},\mathbf{z}) = \mathcal{C}(\overrightarrow{F_u}, \overrightarrow{F_v}, \overrightarrow{N})$ // *form local coordinate system*
18: $T \leftarrow [\mathbf{x}\ \mathbf{y}\ \mathbf{z}]^T$ // *transformation matrix*
19: $G_u^c(s,t) \leftarrow T \cdot \overrightarrow{l_u}$, $G_v^c(s,t) \leftarrow T \cdot \overrightarrow{l_v}$ // *global gradients*
20: $tmp \leftarrow ||G_u^c(s,t) - \overrightarrow{F_u}||^2 + ||G_v^c(s,t) - \overrightarrow{F_v}||^2$
21: $error \leftarrow error + tmp$
22: **end for**
23: Reconstruct G^{i+1} from G_u^c and G_v^c (Appendix A)
24: $i \leftarrow i + 1$
25: **end while**

The parameter *threshold* is predefined to determine the approximation error of the iteration process. In practice a maximum number of iterations can also be specified. After the iteration process is completed, G^i contains the approximated geometry image of the completed gradient images L_u^c and L_v^c. Figure 2 illustrates intermediate results of this iterative progress on the golf ball model.

This process has to be slightly modified for the operation of geometry detail transfer. As the target region has its initial geometry, it is not necessary to perform the geometry initialization described in steps 1 and 2 in the above pseudo-code. In this way, the overall shape of the reconstructed geometry in the target region is still preserved while the geometry details of the source mesh are transferred over.

6.2 Remeshing the geometry image

Once we obtain a geometry image for the hole regions defined on a regular grid, we remesh these regions to meet two conditions: (1) the constructed mesh should fit with the existing geometry connectivity, and (2) it should preserve important geometry features of the synthesized geometry. We perform the remeshing operation in two steps.

In the first step (Figure 3a), we analyze the geometry image to form an *importance map* [1] from which points with high geometry features, which we identify as feature points, are selected for hole regions by employing the method of Ostromoukhov et al. [22]. In the second step (Figure 3b), we use the constrained conformal Delaunay triangulation [26] to triangulate the holes. The input for this operation comprises of hole boundaries and selected feature points. We generate triangles that are similar in size (area) to the triangles of the existing mesh. Note that during the triangulation, more sample points may be added to ensure the Delaunay criterion [7].

7 Patching Holes with Islands

In this section, we discuss how to extend our method to support the situation in where there exist scattered geometry components (or islands) within hole regions. The main challenge is that conventional parameterization methods cannot generate a unique solution for meshes consisting of several disconnected components. To address this issue, we seek to link these disconnected components so that they can be put under a constrained relationship during parameterization. Specifically, we take the following five steps to generate the parameterization for this situation (Figure 4).

(1) The hole is patched (without considering the islands) with a smooth surface that is generated by first linearly interpolating global gradient fields of the existing mesh and then performing geometry image reconstruction using the method explained in Appendix A.

(2) Geometry parameterization is performed on the patched surface.

(3) For each vertex B on the boundaries of the islands, we find a nearest vertex P in the newly patched regions and assign the parameterization value of P to B.

(4) In the 2D parameterization domain, the hole is triangulated based on the boundary vertices of the hole and the islands to form a new mesh consisting of one connected component.

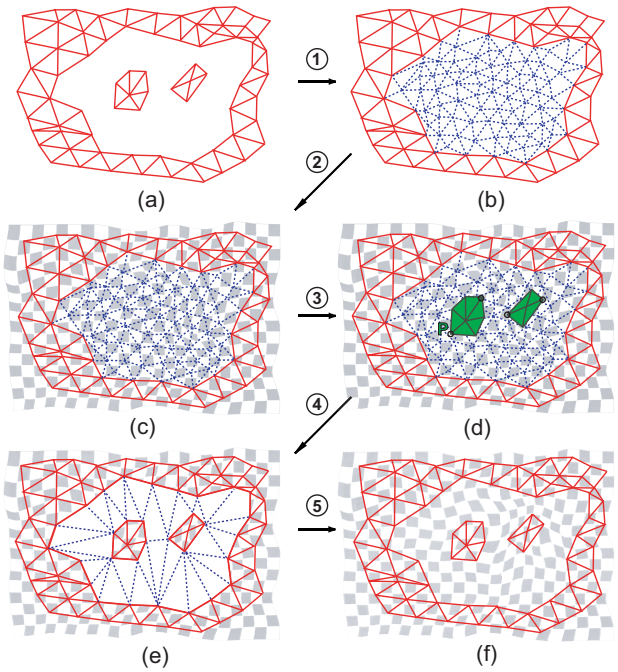


Fig. 4 Parameterization of geometry with disconnected components: (a) a part of a surface with one hole and two islands; (b) the hole is patched with islands removed; (c) the new mesh is parameterized; (d) boundary vertices of the islands are assigned parameterization values of the nearest vertices on new mesh; (e) a simple triangulation is then performed to connect all geometry components; (f) another geometry parameterization is performed with islands included.

(5) We re-parameterize the new mesh. Since the mesh is now equivalent to a connected graph, the parameterization solution is guaranteed to be unique and include islands. From this solution, we generate a geometry image of the existing mesh *only*. We then apply the method explained earlier to patch the hole.

8 Results

We demonstrate our method for filling holes without islands in Figures 5 and 6. In Figure 5, an artificial hole is filled with information adapted from the hole's neighboring geometry. Note that the hair geometry pattern is still preserved in the patched region. Figure 6 shows how features like a sharp corner is also preserved. For comparison, we also patch the holes with smooth geometry, which is generated by first linearly interpolating global gradient fields of the existing meshes and then performing geometry image reconstruction using the method explained in Appendix A. The hole geometry generated by our method appears more convincing than the smooth filling.

Our method for filling holes with islands is demonstrated in Figure 7. Even if there is a big hole, these small islands help preserve important geometry features, especially the two front teeth, and hence lead to better synthesized geometry. Without these islands, the patched geometry appears significantly different from the original mesh.

Table 1 Running time of our method on various models. N_v: number of selected vertices; T_p: parameterization time; T_s: texture synthesis time; T_r: reconstruction time; T: total time.

Model	N_v	T_p(s)	T_s(s)	T_r(s)	T(s)
Igea (Fig 5)	12658	11	28	12	51
Teeth (Fig 6)	4617	3	15	5	23
Teeth (Fig 7)	12841	29	39	23	91

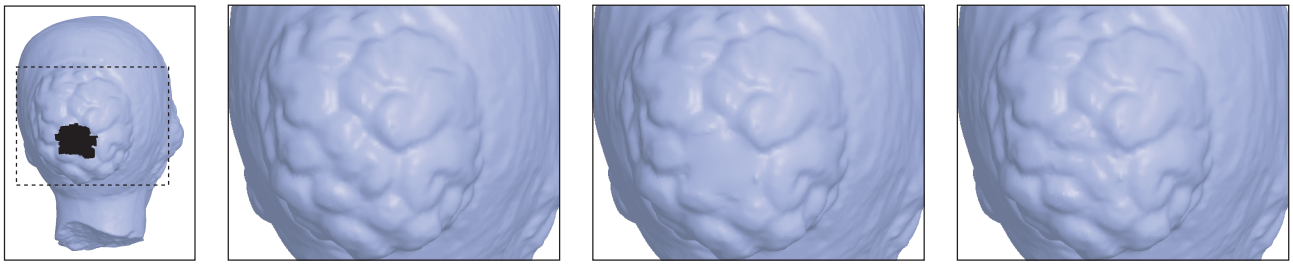
We show a result of geometry detail transfer in Figure 8. The target region (i.e. area to be edited, colored red) is replaced with geometry details from the source region (colored green). In one operation, a deficient part of the mouth region of the Igea model is fixed. Note that the right corner of the lip is synthesized from the left lip corner. In another operation, we grow Igea's hair by synthesizing the existing hair pattern into the marked region. This type of operation resembles a 3D version of "painting by numbers" [14] and image analogies [15], in this context for geometry synthesis.

We report the computation times of our method in Table 1. The computation of our method consists of three main tasks: (i) parameterization of the geometry mesh, (ii) texture synthesis, and (iii) geometry reconstruction. All timings are in units of seconds. All the computation times presented in the table are measured on a PC with a Pentium 4 Xeon 2.8MHz CPU and 1GB of memory. Except for the situation with islands, our overall process time is within one minute. Processing islands takes a longer time due to the additional operations involved for parameterization.

9 Conclusions and Future Work

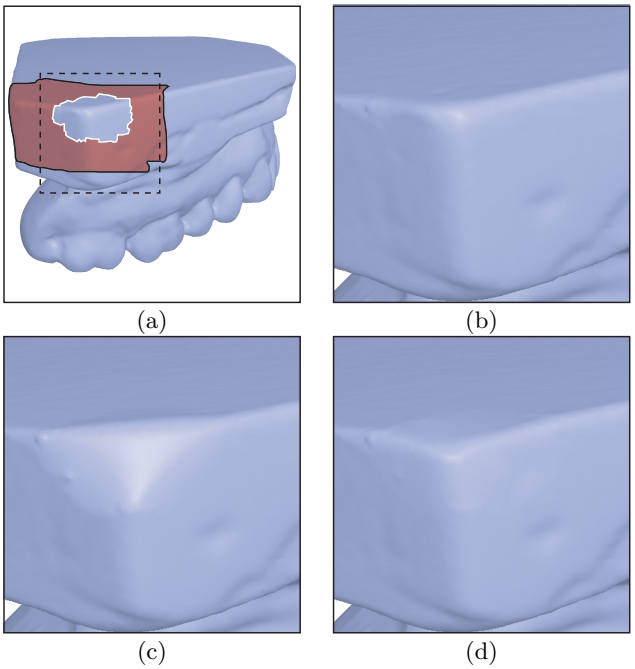
In this paper, we have presented a novel approach to perform geometry completion. In our approach, holes are patched with geometry that not only conforms with their boundaries, but also preserves the geometry details of the original mesh. We facilitate geometry synthesis by converting the working domain from 3D to 2D through geometry parameterization. In this 2D image domain, other well-established image completion techniques can be utilized for more flexibility in control, higher quality and greater efficiency.

There are several research directions that we would like to pursue further. In our current implementation, geometry images of holes are reconstructed from gradient images using a least squares approximation. Therefore, if there are sharp features synthesized in the hole region, our method may not be able to reconstruct the geometry with sharp features preserved. This problem can be



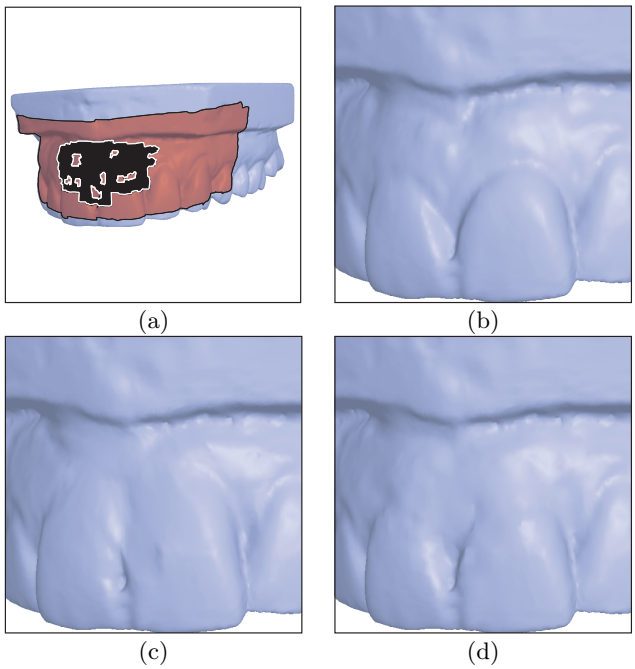
(a) (b) (c) (d)

Fig. 5 Filling holes with existing geometry pattern. (a) A hole is artificially created in Igea's hair. (b) A close-up of the original mesh. (c) The hole is filled with smooth geometry. (d) The hole is completed using our method.



(a) (b) (c) (d)

Fig. 6 Preserving sharp corners when filling holes. (a) A sharp corner of the model of teeth is removed and its surrounding area is selected. (b) A close-up of the original mesh. (c) The hole is patched with a smooth surface. (d) The hole is completed.



(a) (b) (c) (d)

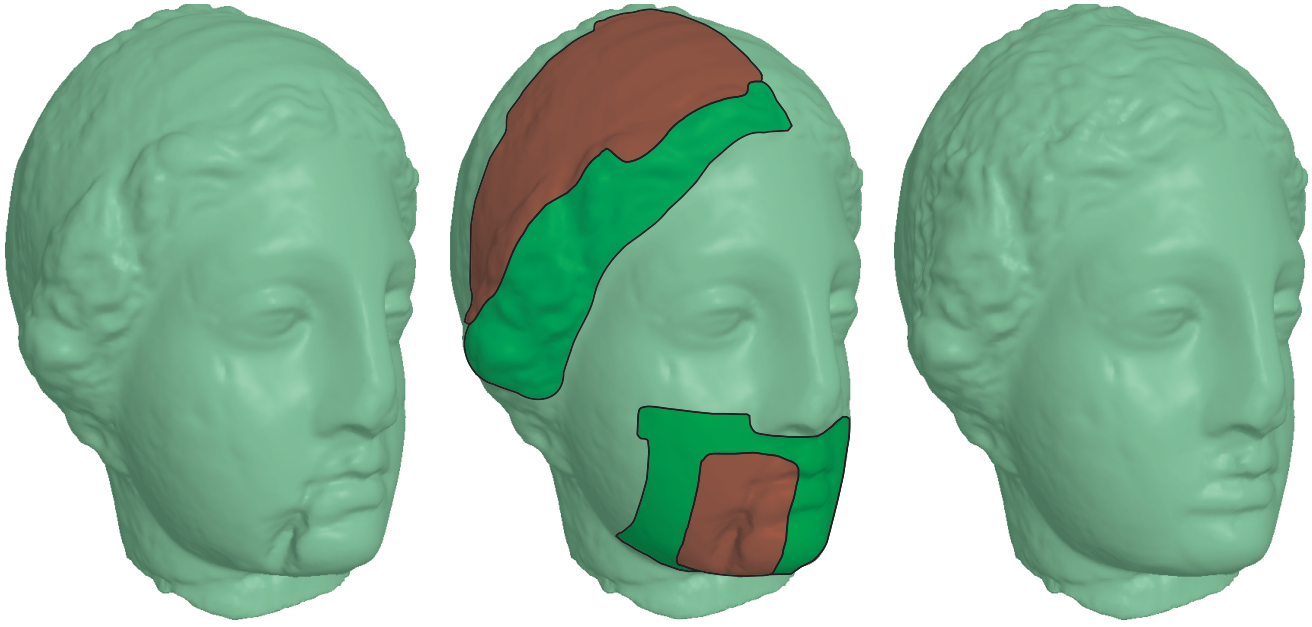
Fig. 7 Filling holes with islands. (a) A hole with several small islands is artificially created on the model of teeth. (b) The original model of teeth for comparison. (c) The hole is patched without considering the islands. (d) The hole with the islands is patched using our method.

alleviated by increasing the resolution of the geometry image (as what is done in Figure 6), but this will result in longer running time for texture synthesis, which is sensitive to image resolution. We plan to further investigate this issue.

Even though we have considered distortion in parameterization during texture synthesis, the results can still be negatively affected by severe distortion, thus affecting the reconstructed mesh. Such distortion is often caused by high curvature surfaces. We wish to make our method less sensitive to this problem by investigating either better parameterization methods or better treatment in texture synthesis.

Finally we plan to extend our method to support point-based models, especially for outdoor scanned data, which is challenging due to the noisiness, incompleteness, irregularity and ambiguity of the input data sets.

Acknowledgements This work is supported by a University of Minnesota Computer Science Department Start-up fund and NSF ACI-0238486 (CAREER). We are thankful to the anonymous reviewers for helpful comments. We thank David Xianfeng Gu for his insightful suggestions and comments. We also thank Lijun Qu for the initial discussion; Hui Xu for the implementation of the preconditioning least squares iterative solver and discussions; and Nathan Gossett and Amit Shesh for suggestions, proofreading and audio production of the video. The golfball model is obtained from the Suggestive Contour Gallery from Princeton, Igea and teeth models are courtesy of Cyberware.



(a) (b) (c)

Fig. 8 Geometry editing. The geometry model before editing (a). The user defines target region (in red) and source region (b). The geometry surface of the target region is then modified by surface details from the source region (c).

A Reconstructing geometry image from gradient images

The problem of reconstructing a geometry image from gradient images can be formally defined as follows: given gradient images $C = (C_u, C_v)$, find a geometry image H such that

$$\nabla H = C \text{ or } \left(\frac{\partial H}{\partial u}, \frac{\partial H}{\partial v}\right) = (C_u, C_v) \tag{A-1}$$

In reality, we may not be able to find H completely satisfying Equation (A-1). Instead, we approximate H by solving a discrete PDE equation with boundary conditions in a least squares sense. H minimizes the following equation:

$$\iint_B F(\nabla H, C) du dv \tag{A-2}$$

in which B is the hole boundary and

$$\begin{aligned} F(\nabla H, C) &= \|\nabla H - C\|^2 \\ &= \left(\frac{\partial H}{\partial u} - C_u\right)^2 + \left(\frac{\partial H}{\partial v} - C_v\right)^2 \end{aligned} \tag{A-3}$$

Equation (A-2) is the same as the equation described in [9]. We employ a similar approach to solve for H, but instead of solving the Poisson equation, we directly convert the equation into a linear least squares problem and then solve it using a preconditioning linear least squares iterative method.

References

1. Alliez, P., Meyer, M., Desbrun, M.: Interactive geometry remeshing. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 347–354. ACM Press, New York, NY, USA (2002)
2. Ashikhmin, M.: Synthesizing natural textures. In: SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 217–226. ACM Press (2001)
3. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 417–424. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000)
4. Bhat, P., Ingram, S., Turk, G.: Geometric texture synthesis by example. In: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 41–44. ACM Press, New York, NY, USA (2004)
5. do Carmo, M.P.: Differential geometry of curves and surfaces. Prentice-Hall (1976). 503 pages.
6. Davis, J., Stephen, Marschner, R., Garr, M., Levoy, M.: Filling holes in complex surfaces using volumetric diffusion. In: First International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 428–438 (2002)
7. de Berg, M., van Kerveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, 1 edn. Springer (1997)
8. Desbrun, M., Meyer, M., Alliez, P.: Intrinsic parameterizations of surface meshes. Eurographics Conference Proceedings **21**(3), 209–218 (2002)
9. Fattal, R., Lischinski, D., Werman, M.: Gradient domain high dynamic range compression. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 249–256. ACM Press, New York, NY, USA (2002)
10. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Trans. Graph. **23**(3) (2004)
11. Gu, X., Gortler, S.J., Hoppe, H.: Geometry images. In: SIGGRAPH '02: Proceedings of the 29th annual confer-

ence on Computer graphics and interactive techniques, pp. 355–361. ACM Press, New York, NY, USA (2002)
12. Gu, X., Wang, Y., Chan, T., Thompson, P., Yau, S.T.: Genus zero surface conformal mapping and its application to brain surface mapping. IEEE Transaction on Medical Imaging **23**(7), 949–958 (2004)
13. Gu, X., Yau, S.T.: Global conformal surface parameterization. In: SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 127–137. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
14. Haeberli, P.: Paint by numbers: Abstract image representations. In: SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques, pp. 207–214. ACM Press, New York, NY, USA (1990)
15. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 327–340. ACM Press (2001)
16. Jin, M., Wang, Y., Yau, S.T., Gu, X.: Optimal global conformal surface parameterization. In: IEEE Visualization, pp. 267–274 (2004)
17. Ju, T.: Robust repair of polygonal models. ACM Trans. Graph. **23**(3), 888–895 (2004)
18. Lai, Y.K., Hu, S.M., Gu, D.X., Martin, R.: Geometric texture synthesis and transfer via geometry images. In: SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling, pp. 15–26. ACM Press, New York, NY, USA (2005)
19. Levy, B.: Dual domain extrapolation. ACM Trans. Graph. **22**(3), 364–369 (2003)
20. Levy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. ACM Trans. Graph. **21**(3), 362–371 (2002)
21. Liepa, P.: Filling holes in meshes. In: SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 200–205. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
22. Ostromoukhov, V., Donohue, C., Jodoin, P.M.: Fast hierarchical importance sampling with blue noise properties. ACM Trans. Graph. **23**(3), 488–495 (2004)
23. Pfeifle, R., Seidel, H.P.: Triangular B-splines for blending and filling of polygonal holes. In: Proceedings of the conference on Graphics Interface '96, pp. 186–193 (1996)
24. Saad, Y.: Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2003)
25. Sharf, A., Alexa, M., Cohen-Or, D.: Context-based surface completion. ACM Trans. Graph. **23**(3), 878–887 (2004)
26. Shewchuk, J.R.: Delaunay refinement algorithms for triangular mesh generation. Computational Geometry: Theory and Applications **22**(1–3), 21–74 (2002). `http://www-2.cs.cmu.edu/~quake/triangle.html`
27. Taubin, G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. In: ICCV '95: Proceedings of the Fifth International Conference on Computer Vision, p. 902. IEEE Computer Society, Washington, DC, USA (1995)
28. Verdera, J., Caselles, V., Bertalmio, M., , Sapiro, G.: Inpainting surface holes. In: Proceedings of International Conference on Image Processing, pp. 903–906 (2003)
29. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 479–488. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000)

Minh X. Nguyen is a PhD candidate in Computer Science from the University of Minnesota, Twin Cities. He obtained his BS degree in Computer Science from Ho Chi Minh City University (now is Ho Chi Minh City University of Natural Sciences), Vietnam in 1994. Before going back to graduate study, he was a lead programmer at a cartography company (DolSoft Co. Ltd., Ho Chi Minh City, Vietnam) specializing in GIS and a junior researcher at the Institute of Applied Mechanics, Ho Chi Minh City, Vietnam. His research interests are interactive 3D visualization with emphasis on hardware support, geometry modeling, scientific visualization and computational geometry.

Xiaoru Yuan is a PhD candidate in Computer Science at the University of Minnesota at Twin Cities. He received a BS degree in Chemistry and a BA degree in Law from Peking University, China, in 1997 and 1998, respectively. His primary research interests include non-photorealistic rendering and its application in visualization, high dynamic range imaging and rendering, volume visualization, illustrative visualization, and computational geometry.

Baoquan Chen is an assistant professor of Computer Science and Engineering at the University of Minnesota at Twin Cities; there he is also a member of the Digital Technology Center and Digital Design Consortium. His research interests generally lie in computer graphics and visualization, focusing specifically on 3D data acquisition, illustrative rendering and visualization and interactive techniques. His research is supported by National Science Foundation, Army Research, Microsoft Research, and private donation. Chen is the recipient of the Microsoft Innovation Excellence Program 2002, the NSF CAREER award 2003, and McKnight Land-Grant Professorship of the University of Minnesota 2004-2006. Chen has served, or is serving on program and paper committees of several conferences in the field, most notably IEEE Visualization (program co-chair 2004, general co-chair 2005/2006), Symposium on Volume Visualization and Graphics (2002/2004), Computer Graphics International (2004/2005), IEEE Volume Graphics (2001/2003), and Symposium on Point Based Graphics (2004/2005, papers co-chair 2006). Chen received an MS in Electronic Engineering from Tsinghua University, Beijing (1994), and a second MS (1997) and then PhD (1999) in Computer Science from the State University of New York at Stony Brook. For more information see http://www.cs.umn.edu/∼baoquan.